HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

Praktikumsabgabe
für Verteile Systeme

REST-API IMPLEMENTIERUNG

im Modul Verteilte Systeme

Studiengang Angewandte Information

der Hochschule Ruhr West

Nils Milewski

10010480

Bottrop, Dezember 2021

Inhaltsverzeichnis

Angabe des ausgewählten REST-API Designs

Da alle Designs ähnlich aufgebaut sind und größtenteils die Darstellung der Vorgaben unterschiedlich ist habe ich mich für das erste Design entschieden.

Begründung der gewählten Programmiersprache und der verwendeten Frameworks

Zusammenfassung

Als eingesetzte Programmiersprache wird Java eingesetzt. Durch die Plattformunabhängigkeit, Robustheit, Sicherheit, Zuverlässigkeit und eine vielzahl an Bibliotheken wird Java bereits in vielen Betrieben eingesetzt.

Aufgrund der aktuellen längeren Unterstützung (LTS) wird Java in der ***Version 11*** eingesetzt.

Die genutzten Frameworks sind ***Spring Boot***, ***Lombook***, sowie ***Junit5***. Das Buildtool bildet ***maven*** und die eingesetzte Datenbank ist ***mongoDB***.

Spring Boot

Als ein Industriestandard gilt ***Spring (Boot)*** für die Bereitstellung von Java basierenden Webanwendungen. ***Spring Boot*** ist eine Erweiterung von *Spring*, die eine komplexe Konfiguration vereinfacht. Der Großteil der Konfiguration wird mithilfe von sog. dependency injection realisiert, bedeutet das ***Spring Boot*** sich im Hintergrund um die Erzeugung und Verwaltung der Objekte kümmert.

Das Framework enthält ein *Tomcat* Webserver, wodurch keine weitere externe Software benötigt, wird zum Bereitstellen

Lombook

Die Bibliothek ***Lombook*** stellt *Quality of Life* Verbesserung bereit die die Entwicklungsarbeit unterstützt ***Lombook*** wird primär im *model* Package eingesetzt, explizit wird die Automatische Generierung von *Gettern/Settern* sowie das *Builderpattern* genutzt

Junit5

Als Testframework wird Junit in der aktuellen Version eingesetzt. Dieses Framework ist der defacto Standard für die Testautomatisierung von Unit-Tests. Junit-Tests werden mit diverser *Annotation* versehen womit die Tests vorbereitet, nachbereitet, manipuliert sowie eingegriffen werden kann

Maven

Für Java existieren diverse Buildtools wodurch die Verwaltung von Bibliotheken, erzeugen von Objektcode sowie Generierung von Ausführbaren code vereinfacht wird. *Gradle* und *Appache Maven* sind beide weit verbreitet, jedoch wird, aufgrund von Vorwissen, ***Appache Maven*** eingesetzt. Als allgemeine Quelle für Bibliotheken gilt *maven central*, hier liegen eine Vielzahl an Bibliotheken als sog. Dependencies bereit. In der *pom.xml* werden alle benötigten dependencies sowie buildmethode beschrieben, *maven* kümmert sich dann darum das alle dependencies vorhanden sind und die entsprechende build schritte ausgeführt werden.

MongoDB

Zur permanente persistierung von Daten wird eine *nosql* Datenbank eingesetzt, hier eignet sich die ***monoDB*** als ein guter Einstieg. MongoDB hält ein Teil der Datensätze im *RAM,* wodurch ein schneller Zugriff auf Datensätze ermöglicht wird. Des Weiteren lässt sich eine ***mongoDB*** sehr gut skalieren und auf diverse Server verteilen. Jedoch ist der größte Vorteil darin das kein festes Schema vorgegeben sein muss was für dieses Projekt gut geeignet ist.

Teststrategie

Mocktio

Damit die Datenbank keine Datensätze der Tests beinhalten werden alle Datenbankabfrage mithilfe von ***mockito*** simuliert. Mithilfe von Annotation ist es möglich die Datenbankanbindung abzufangen und durch von ***mockito*** definierten Resultate zu ersetzen.

Bsp. Folgender Code soll das Löschen aller Datensätze illustrieren, wo mockito jedoch eingreift und das Löschen nur simuliert.

```
@Autowire
private UserRepositor repo;
@Test
void run(){
      doNothing().when(repo.deleteAll());
      repo.deleteAll();
```

Unit Tests

Die Teststrategie dreht sich primär um Unit Tests, welche die jeweiligen Controller und Endpunkte testen sollen.

Die Testklassen sind mithilfe von Package nach Endpunkten gruppiert. In jedem Package sind drei Testklassen die jeweils *POST*, *PUT* und *GET* Zugriffe abdecken, der *DELETE* Zugriff wird nicht explizit getestet da der Endpunkt nichts zurückgibt.

Da jeder Endpunkt ungefähr gleich aufgebaut ist von den Zugriffen her konnte man diese mithilfe von abstrakten Klassen verallgemeinern. Insgesamt gibt es drei Abstrakte Klassen die jeweils *PUT*, *POST* und *GET* Methoden samt Fehlerfälle abdecken und die Klasse AbstractTestData erweitern.

Als allgemeiner Datencontainer steht die Klasse AbstractTestData, hier werden für die Tests jeweils eine Liste sowie ein Objekt zur Verfügung gestellt, beide liegen auch als leere Objekte.

Eventuelle Limitierungen

Allgemeine Limitierung stellt die vorhandene Hardware dar.

Ein Nachteil von Spring ist die schlechte Skalierbarkeit.

Anhang

Source code

Der Ordner ***src*** bildet das Stammverzeichnis für das Projekt und liegt als ZIP Archiv vor.

- test: Enthält die genutzte Testklassen für das Projekt
- main
  - /java/…: Speicherort der Quelldateien
  - /resource: Verzeichniss für genutzte resource files

Da git als Versionsverwaltung eingesetzt wurde liegt der Sourcecode ebenfalls komplett auf GitHub in einem privaten Repository zur Verfügung, auf Wunsch ist eine Einladung oder Öffentlich Schaltung von diesem möglich.

GitHub link: https://github.com/nimile/Distributed-Systems-Practical-Task

JAR File

Die Datei … ist das Ausführbare Projekt, zum Ausführen sind die Hinweise der *readme.pdf* zu beachten

Nutzungshinweise

Die Datei *readme.pdf* beinhaltet eine Anleitung zum Ausführen des Projektes sowohl in *IntelliJ IDEA 2021* als auch als eigenständige Ausführbare Datei. Des Weiteren sind mögliche Probleme und Ihre Lösungen hier beschrieben.

Dokumentation

Die Datei *Dokumentation.pdf* ist dieses Dokument

Verzeichnis /Swagger

Im verzeichniss */swagger* liegen der sourcecode für die swagger Dokumentation samt HTML code welche mithilfe von swagger generiert wurde.

Verzeichnis /javadoc

Die Dokumentation des Quellcodes befindet sich als javaDoc unter dem Verzeichniss */javadoc*

Verzeichnis /Word

In diesem Verzeichnis liegen alle Word Dateien, die für die Dokumentation genutzt wurden